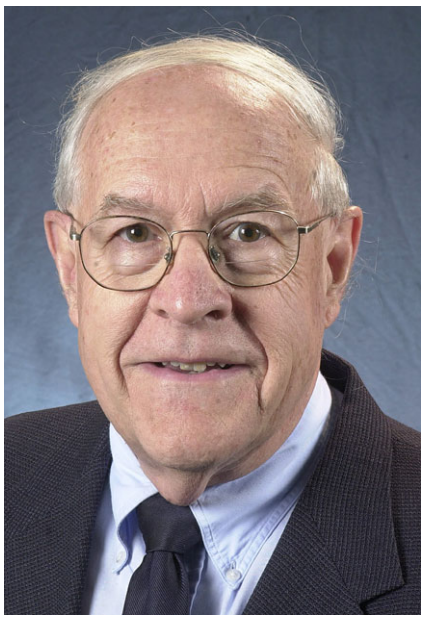
Interview with Fred Brooks on "Building Effective Large-Scale Requirements"

The Design Science perspective remains by no means undisputed. One of its most outspoken critics is Turing Award Winner Fred Brooks, manager of one of the largest early software projects in the 1960's and author of some of the most influential early software engineering studies, including his classic book "The Mythical Man Month". Frederick Phillips Brooks, Jr. is a software engineer and computer scientist. He received the A.B. in Physics from Duke University in 1953, and the Ph.D. in Computer Science from Harvard University in 1956. Working with the IBM Corporation from 1956 to 1965, he was an architect of the Stretch and Harvest computers and then was Project Manager for the development of IBM's System/360 family of computers and OS/360 software. In 1964, Brooks founded the Department of Computer Science at the University of North Carolina at Chapel Hill and chaired it for 20 years. Currently, he is Kenan Professor of Computer Science.

DOI 10.1007/s12599-010-0104-x



Prof. Frederick P. Brooks, Jr., Ph.D.

Interview by

Sean Hansen
The Weatherhead School
of Management
Case Western Reserve University
10900 Euclid Avenue
Cleveland 44106, OH
USA

Prof. Kalle Lyytinen Ph.D.
Iris S. Wolstein Chair
The Weatherhead School
of Management
Case Western Reserve University
10900 Euclid Avenue
Cleveland 44106, OH
USA
kjl13@case.edu

Prof. Dr. Matthias Jarke (✉)
Lehrstuhl für Informatik 5
RWTH Aachen University
Ahornstr. 55
52056 Aachen
Germany
jarke@dbis.rwth-aachen.de

Published online: 2010-05-05

This article is also available in German in print and via http://www.wirtschaftsinformatik.de: Hansen S, Lyytinen K, Jarke M (2010) Interview mit Fred Brooks zum Thema „Erhebung effektiver Anforderungen im großen Zusammenhang". WIRTSCHAFTSINFORMATIK. doi: 10.1007/s11576-010-0226-2.

Lyytinen: One of the reasons we wanted to converse with you in this special issue on requirement engineering (RE) is that much of the RE literature makes an assumption that fixed requirements are absolutely vital for design success. We know that you are critical of some of the assumptions underlying this model. So why do you think that it is a problematic model?

Brooks: In my experience, two things happen. One is that requirements change during the process of building any complicated system, because the outside world changes. Secondly, one discovers requirements that one did not know existed even at the beginning of a project. I often quote from a study report done by an Air Force Science Board Committee on requirements. They say that the notion that you need to have your requirements done before you do Milestone A is wrong. What you really want to do is to set your overall goals by Milestone A and then develop the requirements by Mile-

stone B, as you do the design. The problem comes not so much from attempting to get requirements as from the notion that you can develop *a complete and binding list* of requirements.

In the classical Waterfall Model, there is only one level of feedback from each next level down. Thus if you follow that process – and I have been through it several times – you come out with a list of requirements that nobody knows how much it will cost to meet – cost in performance, or cost in dollars, or cost in schedule. All the stakeholders pile requirements on at the start. The Air Force Study Board report was very interesting in one respect. It pointed out that it is now taking 15 years to develop a substantial military system; 25 years ago it was taking 5 years, and the difference is the process. In those five-year systems, they started with some very firm goals and very firm leaders, but not with very firm detailed requirements. Those were worked out during development.

I am an advocate for Barry Boehm's spiral model, in which you do some feasibility analysis, you do some design, you do some requirements development, you iterate, and you test things on each iteration. You build prototypes as soon as you can.

In the literature, unfortunately, prototyping is seen principally as a means of evaluating user interface requirements, whereas in fact prototyping is important for evaluating function as well.

Hansen: The insight from the Air Force Study Board regarding the dramatic extension in the timeline is quite striking. Is your perception that we have moved backward by putting formal process in place?

Brooks: The whole agile movement starts with the assumption that our development processes got much too heavy and too complicated, and therefore too slow; that we wanted to move to a model in which one builds stuff early and tests it continually – not *continuously*, *continually* – with use and user feedback. I think that is exactly right. Harlan Mills proposed this back in 1971. Consider a real-time system where you have a cycle of some sort. You want to build that basic backbone and "stub" out all the functions. Then one by one put function in, try it on a user, refine it, and then flesh out another stub. As soon as you have something useful to the user, really deliver it and really start getting feedback from the field. That is also essentially what the agile movement is now saying.

Lyytinen: To what degree do you think that this an outcome of the fact that, because of cost reasons and many other things, you have very formal tendering and contracting processes and in order to do that you have to have so-called full requirements specifications?

Brooks: I think that is a big part of the problem. Indeed, I have a chapter in my book, *The Design of Design*, coming out this month that is called "Requirements, Sin, and Contracts". The argument is that people need formal contracts to protect them from cheating each other, intentionally or unintentionally, and the formal contracts lead to stating requirements too soon. Then, that may lead to a strategy on the part of the unscrupulous to low-ball on the initial formal contract and then make it up on the change orders. I notice that in buildings that is not what is done. You give the architect a rough set of goals. The architect comes back with a program, which is essentially a statement of requirements. That is iterated with the user. Then the architect does a conceptual design that is iterated with user, using drawings and models. One does not let the construction contract until the design is complete. I think the mistake we make often in software engineering is insisting that the contract be let before there is a design. That is a serious error.

Hansen: You mention that we freeze the requirements too soon. Do you think there is a "right" timeline for freezing?

Brooks: No, I do not believe there is a point at which you have a complete and final set of requirements. I once worked on a central payroll program for 40 states. Just because of what the 40 state legislatures were doing, the externally imposed requirements were changing all the time. If we look at many software systems that have various kinds of input-output devices, the technologies change all the time – the world keeps changing around you. It is crucial to have a process that is flexible to change in requirements. It must also respond to costs and difficulties encountered in development.

The other factor is that for any successful system the scope of usage is going to be enlarged way beyond what you expected. Thus there will be added requirements as people adapt the system to new functions. They will discover that to adapt it really neatly it ought to have some other capabilities. For example, as we discovered with building a new Computer Science building, lo and behold it accommodates really nicely conferences up to 125 people without interfering with normal activities. In any modification to the building it has newly become a requirement not to undo that capability. So the process of changing requirements goes on not only in the entire development process but for the entire product lifetime. The concept of a fixed requirement time is a gross approximation that misleads us.

Lyytinen: So, how much do you have to pay attention to the fact that requirements may change over time? One could also argue that sometimes this change is outside of your control. There are many drivers, which influence the level at which you have to pay attention?

Brooks: An experience I had with a house design is that we discovered a requirement quite later on, while running use cases against a set of drawings – a requirement that we had never thought about before. We have meetings of 40 people here – where do they put their coats? We had not thought about that one. It turned out to be a little trigger on a big decision that already had many pros and cons. That requirement caused us to flip the whole house end for end. Well, that was quite a surprise. You might say "You should run all possible use cases ahead of time", but you *cannot* do that. As one designs, one starts looking at prototypes – such as drawings or physical models or Wizard of Oz prototypes (I am taking "prototype" to be very abstract and general here). As you start running more and more detailed use cases against them you discover things that you need that you did not realize you needed.

Sooner or later, if you are going to enter a fixed price contract, you do have to settle on a set of specifications, but notice that, as I say, in the building industry that fixed set of requirements is prepared *after* the architect has completed the design.

Lyytinen: Another danger that some people point out is requirements paralysis. That you just do too much requirements specification. Are there examples where you might be investing too much time on requirements?

Brooks: Well, I have seen two cases. With OS/360, the marketing organization put together a requirements group and they came up with a long and ridiculous list. As project manager I just had to finally say, "We are not going to do that."

In another case, when I was on the Defense Science Board I was called on to review, along with a Marine general who had spent his life in military aviation, the plans for a light attack helicopter. While a colonel was briefing us at the Pentagon he said without changing his tone of voice, "And it has to ferry itself across the Atlantic." *It what?* I do not know anything about helicopter design, but I know enough about design to know that you cannot do that without giving up some other attributes. Naturally the General and I both asked, "Why does it have to do that?" Because, when you think about it, a helicopter only has do that twice in its life if you are lucky, and once if you are not. The answer was, "We do not have enough C-5 transport aircraft to carry them all." So we said, "Well, why not take some of the total program money and buy a few more C-5s?" "Oh we cannot do that" was the response. In other words, that was not *bureaucratically* possible. So somebody had put this unfortunate design requirement on the helicopter. (The project eventually died.)

Then we inquired about the requirements process. There had been a committee representing the different user communities. As far as we could determine there was neither a helicopter engineer nor a pilot on that committee. So who were they? There were representatives of the various user organizations, in other words, fundamentally paper-people. They had talked to their constituencies; each constituency had stated what they wanted. There was the usual log rolling – "I will not naysay your requirement if you will not naysay my requirement." So consequently nobody had said, "This requirement does not make any sense. There is bound to be a better solution to how to get them to Europe than that." That was a very vivid example to me, even though it was outside my area of competence, of how the requirements process can get detached from the design process and lead you to this long, long list of unreasonable requirements.

Well, I think our industry has somewhat moved away from that notion of a long list of pre-specified requirements – that you do all that before you start design – at least I hope it has. So the question of "Is it possible to spend too much time and too much effort getting requirements before you do anything else?" can be answered with YES. Now, does that mean that you do not do as careful an elicitation of initial requirements as you can? Of course not. Surely you do that, but you have to have a process that assumes that the process is not absolutely right, and absolutely not final.

Lyytinen: Based on your experiences, what are the biggest challenges in capturing and managing the requirements so that they are really useful and sensible for many different stakeholders?

Brooks: I think the biggest challenge is getting a unified overall vision of what it is you are trying to do, as opposed to many fragmented personal visions. So, your next question is then, "How do you do that?" And frankly, I do not know a general solution. That is a chief function of a project leader.

Lyytinen: How did you try to do that yourself with the IBM/360 project?

Brooks: Well, I had two different System/360 experiences of course – the hardware one and the software one. In the hardware case, the Spread committee spent six weeks in a motel in Connecticut with representatives from every division of the company – marketing, the military systems division, the World Trade company, as well as the different divisions that built I/O gear and processors, etc. What we hammered out was a vision of a single new product line replacing the six that we had. Those were all running out of architectural gas, namely address space. So, as part of hammering out the product strategy, we developed a technical vision. As a matter of fact I wrote the document of 40 technical ground rules for the new computers that summarized our vision. It was a short document – two or three pages for 40 rules. Now, we understood that the requirements were to cover the entire computer performance and configuration range from little to big, with strict program compatibility, so that was the first overall objective. The next overall objective was that the system had to be effective across all different applications. Another crucial one was we were going to make this out of a technology that cost about half as much and we needed to keep the company revenue at least where it was, and indeed, grow it. So, we had to be able to create new applications and markets to more than double the amount of computing that people wanted. Those are quite clear objectives.

With the software, we had the opportunity because of the compatible hardware to say we can build *one* software package to support all of the System/360 hardware systems and configurations. We did not achieve that completely, but we did go a long way towards it. We knew both the existing applications and the new communications-based applications that we had to satisfy. We knew the languages we had to support – ALGOL, COBOL, FORTRAN, report program generator, assembler, etc. We knew that the operating systems had to support a new vision – that the operating system is in control rather than the human operator. That was an entirely new vision; nobody had really done that before. We also planned to provide multiprogramming, another new capability. So the overall vision was a disk-based system, a multi-programmable system, an application-broad system, a language-broad system, a performance-broad system configurable for different memory sizes from 16 K up to 500 K at least.

Thus we had a few very clear overall broad objectives and we tried to keep those forefront, just as the Air Force committee described for successful weapon programs. At one point, when the programming house was entirely separate from the System/360 development project, the programming house came back with a software support plan that had four different incompatible operating systems. You would have to recompile your program as your configuration grew and you changed memory sizes or processors, even though they were strictly binary compatible. From an overall project point of view I said, "That is unacceptable! The overall vision was one of smooth customer growth; this violates the overall vision." We scrapped a year's worth of work and started over. But that is the advantage of having very few, very clear overall objectives. Those constitute a unified vision.

Lyytinen: Is there anything new that we have to do now compared to what you had to do in earlier development projects?

Brooks: I think we have a clearer perception that the tail is *long*. When we were doing the /360 hardware, we predicted that the architecture would last 25 years. We are now at 45 years today, and System/z is newly out there and still executing an upward version of the same architecture. I thought I was being bold in estimating 25 years, because of course the implementations turn over every four to five years. The notion that an architecture has to last and last is now clearer for us. For example, look at the current FAA air traffic control system. It has been in place

since the early 1960s and they are still designing the next one.

Hansen: Why do think that is the case? Is it a question of path dependence?

Brooks: The effort of re-doing a system is so much bigger than expected. The effort is not just the cost of redoing, it is the cost of retraining everybody that is going to use it or maintain it. I was interested to find out that as recently as five years ago there was still an IBM vacuum tube 650 in productive service in Germany. Well, you know it was doing a job. So the value has just persisted. The lifetimes we are talking about are much longer than we expected, and the maintenance tail of a successful project is much longer than we expected. I think that is a realization that has dawned on us bit by bit.

I think another one that has dawned on us – well, I remarked on it in 1975 in "The Mythical Man-Month", but I think we appreciate it more and more – and that is the notion that a successful product attracts new applications and the new applications in turn call for added function and added performance. So, this virtuous cycle with a successful product can go on and on. Now, it can go on and on ridiculously. Look at Microsoft Word and the number of different options and functions in it, which is ludicrous. It is not clear that what improved over the years is ease of use.

Lyytinen: Do you have any closing thoughts?

Brooks: The most important single message I can leave you with is that great designs come from great designers and not from great processes. What process improvement does – and it is important – is bringing up the *floor* of practice. It enables organizations by systematic methods and better processes to improve poor practice towards average practice. What process work does not do is to raise the *ceiling* of project imagination and quality. Ceiling raising comes from really gifted people who are given opportunities and *authority* to do great designs. I have investigated IS designs that have fan clubs. Which of these emerged from normal product processes, and which were developed outside of normal product processes? The striking thing is that most of them were outside normal product processes. Such processes are designed for development of follow-on products; they work well for that.

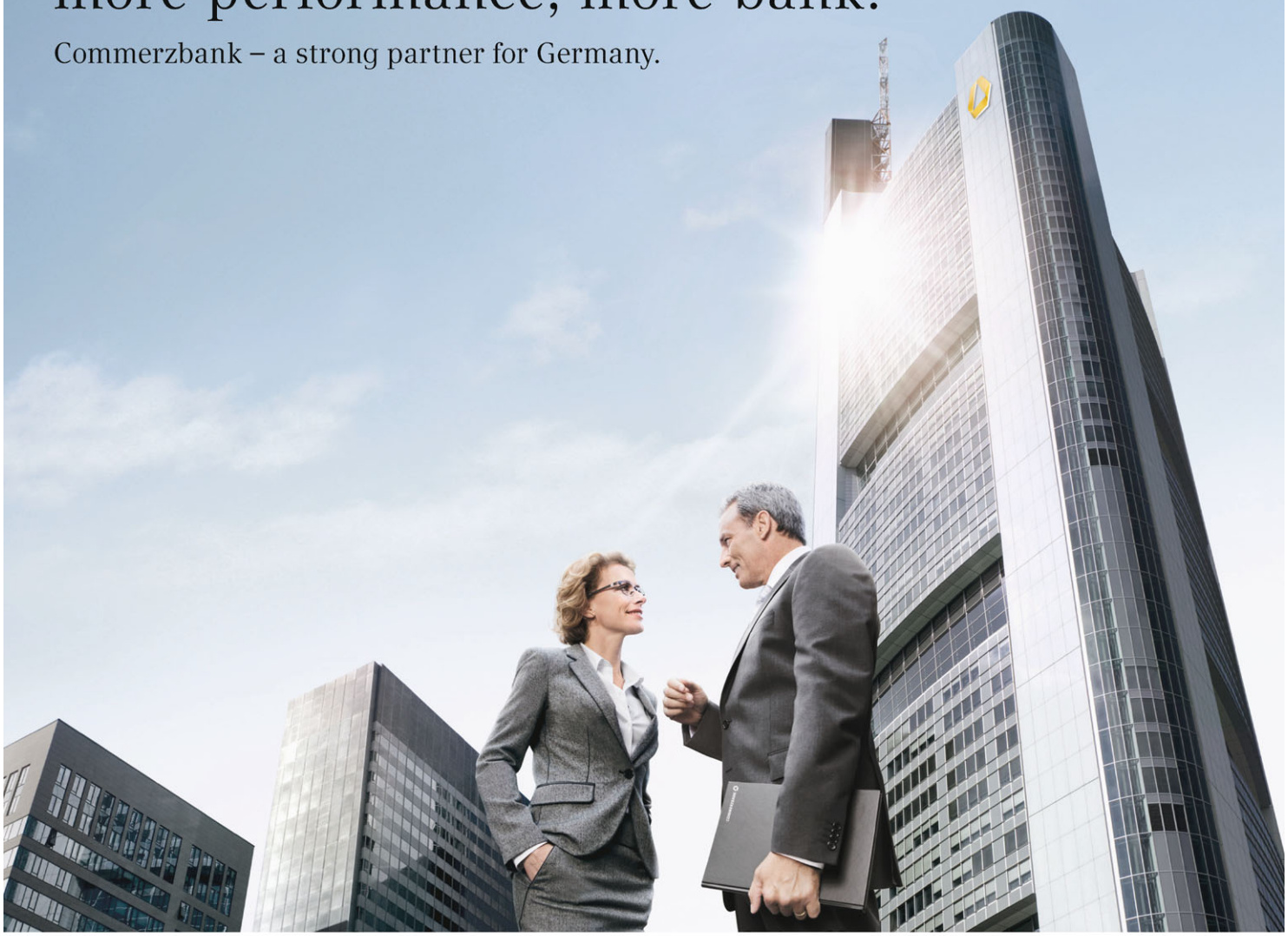
That observation means that the most important thing that a manager can do is to find potential great designers and grow that talent thoughtfully and systematically. Thus I would go back to a focus on people and talent, as opposed to process. That is from a management point of view, rather than a research point of view.

Lyytinen and Hansen: Thank You.

COMMERZBANK

The new Commerzbank: More partnership, more performance, more bank.

Commerzbank – a strong partner for Germany.



The new Commerzbank unites the strengths of Commerzbank and Dresdner Bank. As the leading bank for retail and corporate clients in Germany, we combine strong local links and a long tradition with the end-to-end expertise and power of an international network. In this way, we live our core values of partnership and performance: in our dealings with clients, investors and employees, and in providing outstanding quality and service. This makes us a reliable, strong long-term partner – Germany's bank of choice. **www.commerzbank.com**

Achieving more together

Gehen Sie den Fragen der Zukunft auf den Grund!

Eintauchen 2010 – das McKinsey-Seminar für Doktoranden und Studenten, die sich für Business und Technologie begeistern. Vom 10. bis 13. Juni 2010 in Lissabon.

Welche Strategien brauchen IT-Unternehmen, damit sie und ihre Kunden den Anforderungen aus Klimawandel und steigendem Umweltbewusstsein gerecht werden? Wie kann Green IT helfen, die Herausforderungen der Zukunft anzugehen? Gewinnen Sie in einer spannenden Fallstudie Einblicke in die vielfältigen Tätigkeitsfelder des Business Technology Office. Lernen Sie die Menschen kennen, die McKinsey ausmachen, und erfahren Sie alles über Ihre persönlichen Einstiegsmöglichkeiten und Karrierechancen. Bewerben Sie sich bis zum 25. April 2010 unter www.eintauchen.mckinsey.de

McKinsey&Company

IN A CLEAR NEW WORLD
OPPORTUNITIES HAVE NOWHERE TO HIDE

The next big thing is out there. Which is why the best-run businesses today are those that can spot it first. It's also why so many companies are turning to SAP® business software. They're using it to uncover once-hidden nuggets of opportunity and find new ways to become more agile. And they're discovering that the next big thing isn't always "out there," but has often been in front of them the entire time. It's how business gets done in a clear new world. **Start the journey. Visit sap.com/clear**

THE BEST-RUN BUSINESSES RUN SAP™

SAP®

Deutsche Telekom Laboratories

We shape the future

Deutsche Telekom Laboratories are Deutsche Telekom's research and development institute. Established under private law, this scientific facility is an affiliated institute of the renowned Technische Universität (TU) Berlin. At Telekom Laboratories, some 350 scientists from around the world and experts from the Deutsche Telekom Group develop new, innovative services and solutions for the Group's customers. The founding of new spin-off companies is another way the Group utilizes Telekom Laboratories' results.

Cooperation with the TU Berlin, other universities and industry partners creates a bridge between business and science in order to turn ideas into marketable innovations as quickly as possible. Telekom Laboratories' innovative processes are based on "open innovation" principles that enable the free exchange of ideas and information between selected institutions and companies. The objective is to capture synergy effects and to continue enhancing research results, as quickly and efficiently as possible, through interchange. This also encompasses the inclusion of users and customers in the innovation process, as enabled by a number of inventive methods.

As part of its activities, Telekom Laboratories focuses on five fields of innovation (5i):

- Intuitive Usability of services and devices
- Integrated Service Components
- Intelligent Access
- Infrastructure for IT and telecommunications
- Inherent Security

The business and information systems engineering offers useful interdisciplinary approaches for all these areas of innovation. Subject matter includes, for example, modeling, methods and tools for process innovations, agile architectures for information and communication technologies (ICT), technology-oriented management approaches and techno-economic assessments. The aim is to safeguard the economic sustainability of innovations for the Group.

Telekom Laboratories is divided into two areas: The Innovation Development Laboratory focuses on near-market developments with a time horizon of 18 months to three years. In the Strategic Research Laboratory, scientists holding seven professorships work on long-term technology and applied research.

Aside from its Berlin headquarters, Telekom Laboratories also operate facilities in Darmstadt (Germany), Beer Sheva (Israel) and Los Altos (United States).

Contact:
Deutsche Telekom Laboratories
Ernst-Reuter-Platz 7, 10587 Berlin
E-mail: wi.laboratories@telekom.de
www.laboratories.telekom.com

We proudly present

Business & Information Systems Engineering (BISE) is the new peer-reviewed scholarly e-journal for the entire techno-economically oriented community with a focus on design science-oriented research. It continues the 50 years' tradition of the journal WIRTSCHAFTS-INFORMATIK by that all articles appear both in English and in German. Moreover, authors benefit from our double-blind, constructive, and rapid review process.

„I believe the time is opportune for the IS community to provide such outlets for design researchers. BISE, however, has some unique advantages for positioning itself as a desirable outlet for design science research."

Alan R. Hevner

www.bise-journal.org